IE1204 Digital Design

L10: State Machines (Part 2)

KTH Informations- och kommunikationsteknik

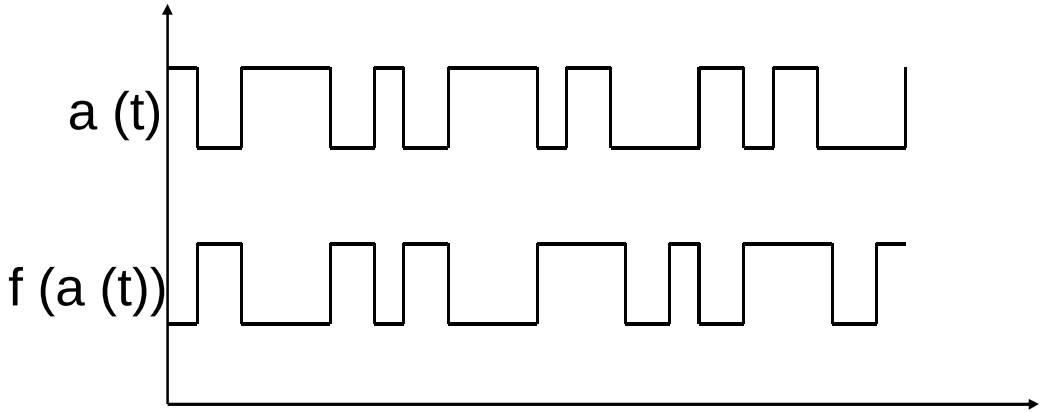
Masoumeh (Azin) Ebrahimi (masebr@kth.se)

Elena Dubrova (dubrova@kth.se)
KTH / ICT / ES

This lecture

- BV pp. 528-532, 557-567

Sequential System



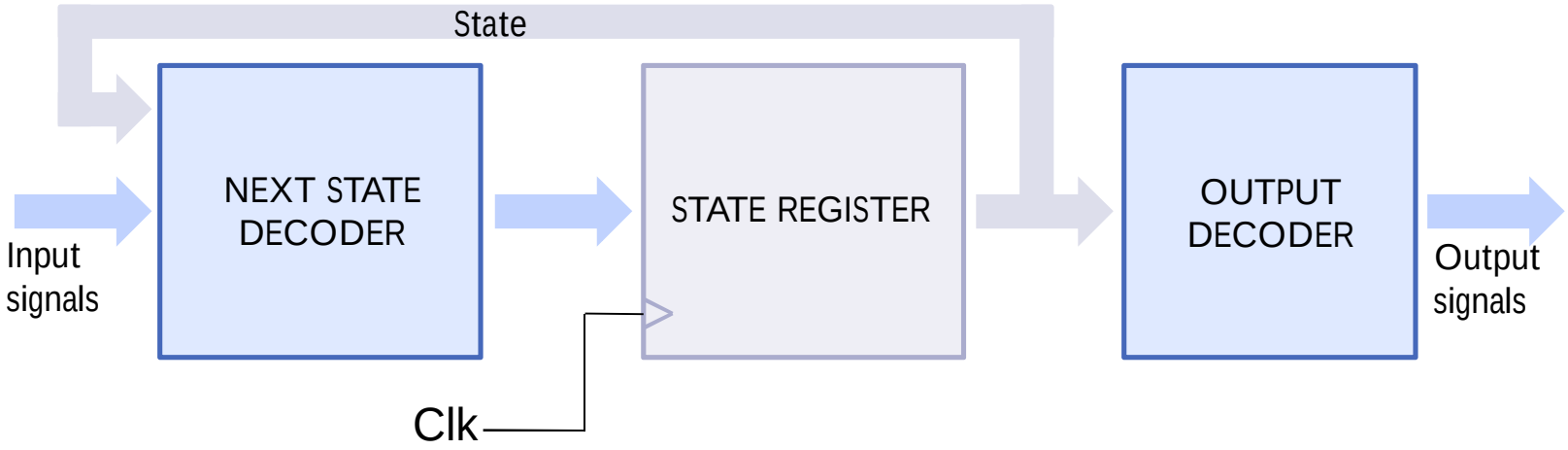
A sequential system has a built-in memory - the output depends therefore BOTH on the current and previous value(s) of the input signal

Lecture 8 - Lecture 13

Basic method for the design of state machines

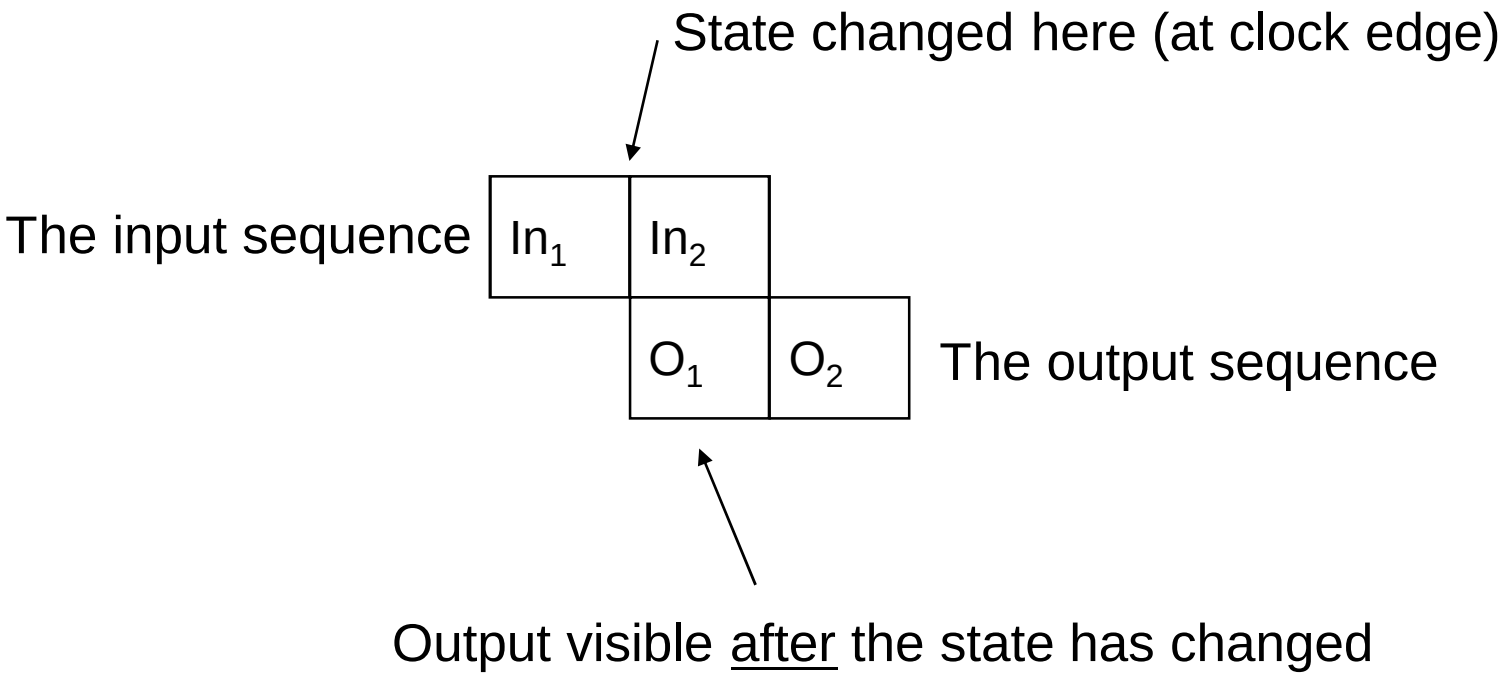
1. Analyze the specification of the circuit
2. Create state diagrams
3. Set up the state table
4. Minimize state table (this lecture)
5. Assign codes for states
6. Choose the type of flip-flops
7. Realize the circuit using Karnaugh maps

Moore Machine

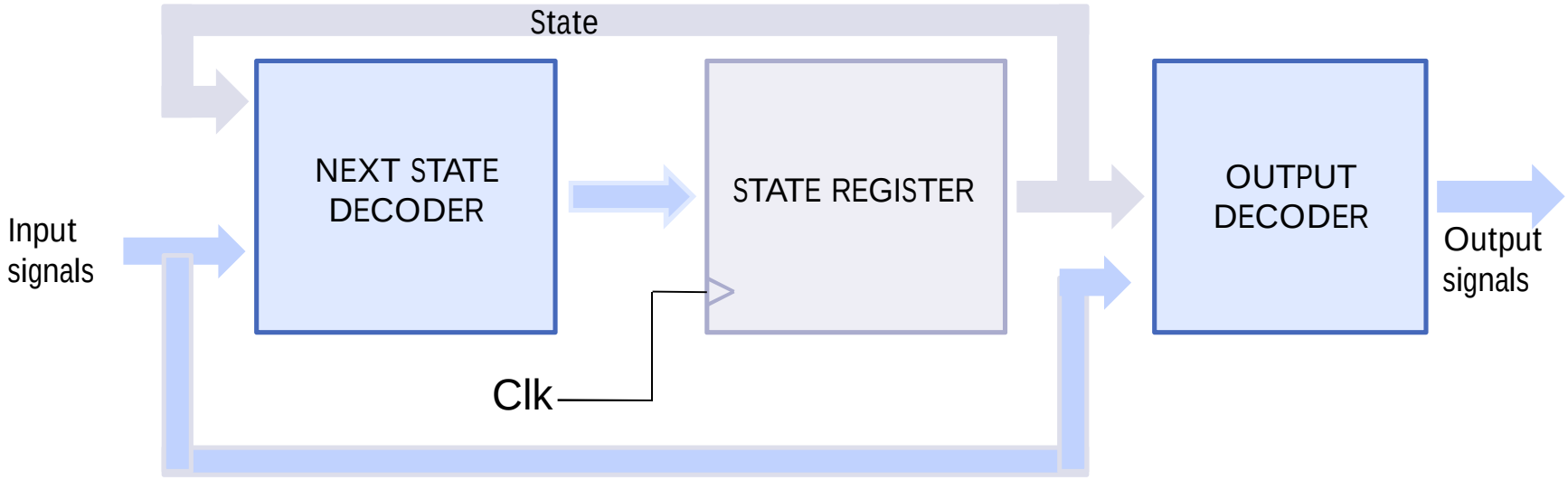


- In a Moore-type machine output signals depend only on the current state

Input vs. output - Moore

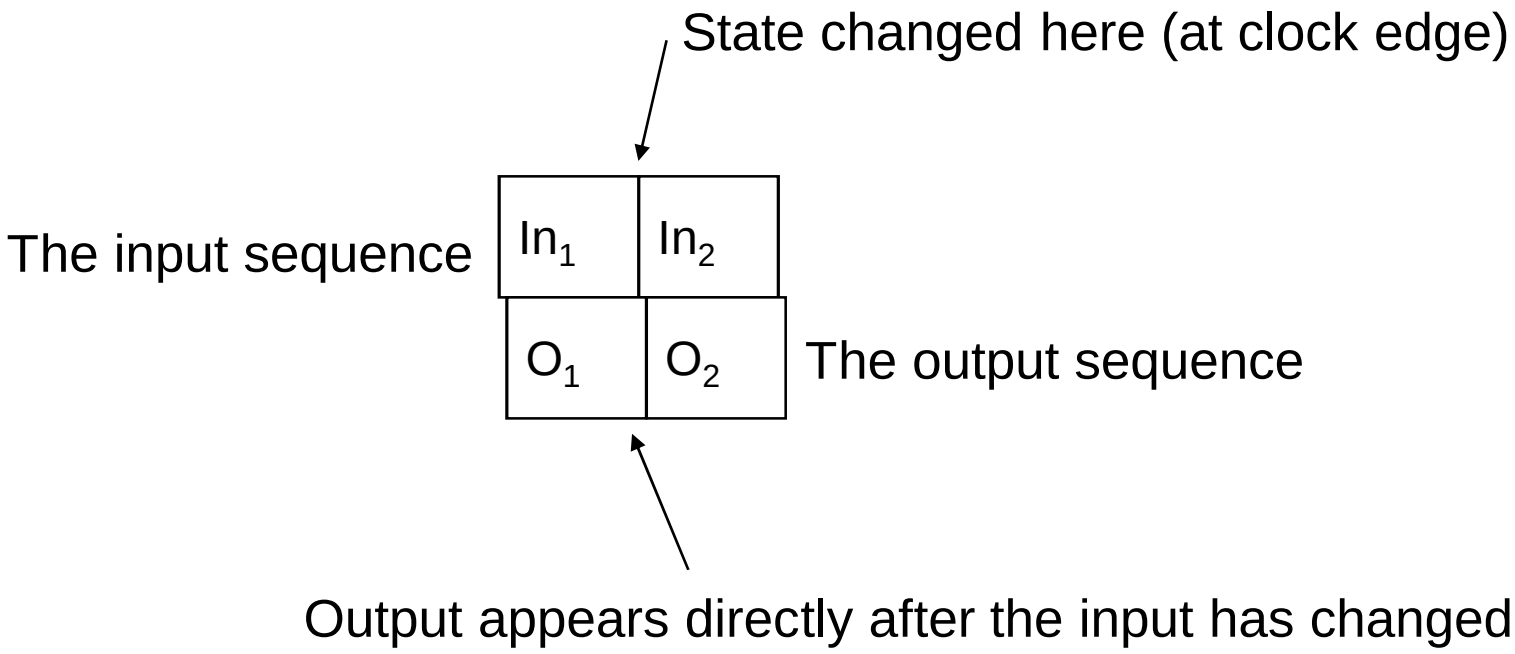


Mealy-type machine



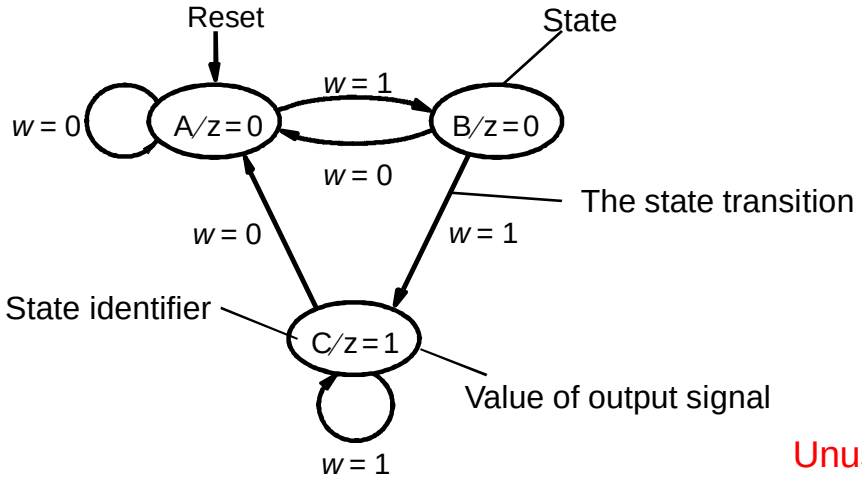
- In a Mealy machine, output signals depend on both the current state and inputs

Input vs. output - Mealy



Unused state

- Sometimes you get more states than you need when selecting a code
- ”Unused” states must be taken care of so that the state machine does not hangs at the start-up (if reset is not used)

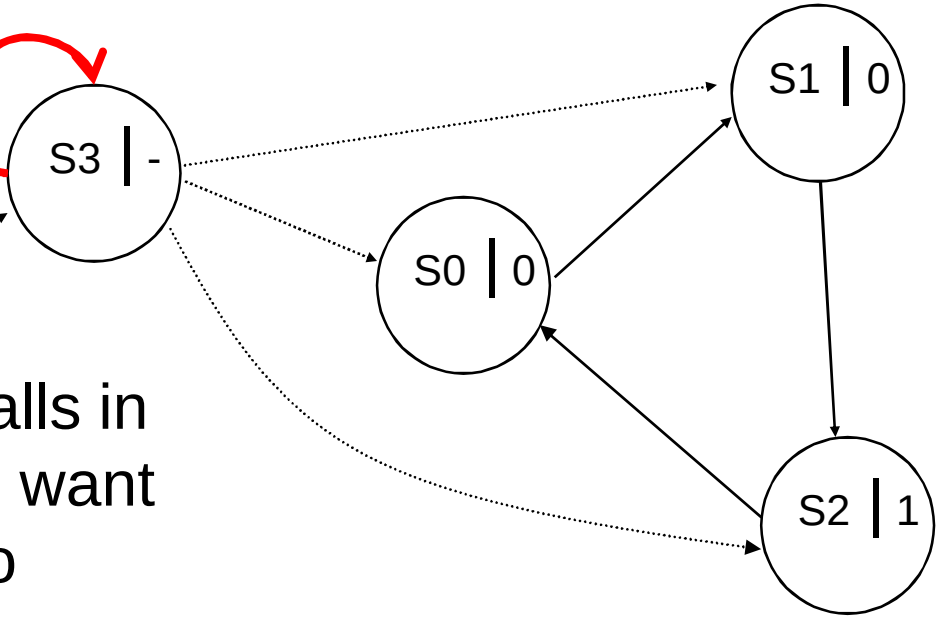


	Present state	Next state		Output
		$w = 0$	$w = 1$	
	y_2y_1	Y_2Y_1	Y_2Y_1	z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
Unused State	11	*dd*	*dd*	*d*

Example: (0,0,1) sequence generator

3 states => 2 flip-flops. One unused state.

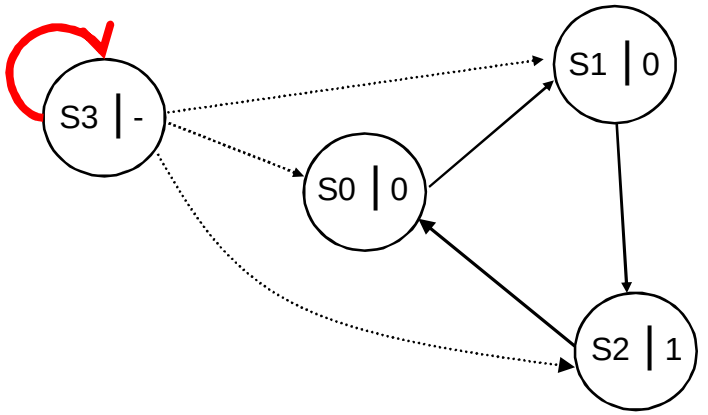
Dangerous transition (Machine hangs)



If the machine falls in this position, we want it to find move to another state as soon as possible

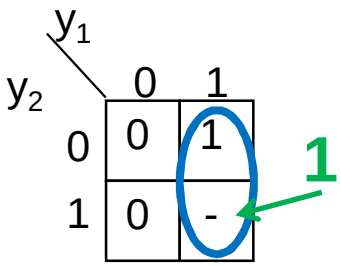
Next-state function

	Current state	Output signal	Next state
	y_2y_1	z	Y_2Y_1
S0	0 0	0	0 1
S1	0 1	0	1 0
S2	1 0	1	0 0
	1 1	-	- - (not 11)

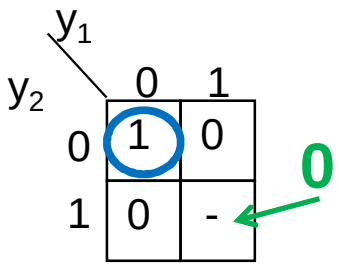


Karnaugh maps

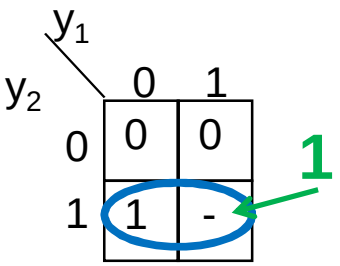
	Current state y_2y_1	Output signal z	Next state Y_2	Y_1
S0	0 0	0	0	1
S1	0 1	0	1	0
S2	1 0	1	0	0
	1 1	-	-	- (not 11)
	1 1	1	1	0



$Y_2 = y_1$



$Y_1 = \overline{y}_2\overline{y}_1$

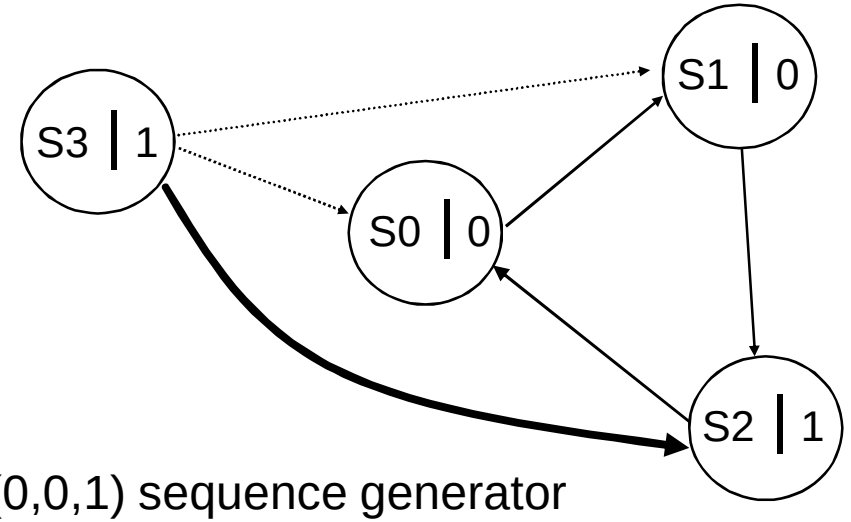


$z = y_2$

State table after Karnaugh minimization

	Current state	Output signal	Next state
	y_2y_1	z	Y_2Y_1
S0	0 0	0	0 1
S1	0 1	0	1 0
S2	1 0	1	0 0
	1 1	1	1 0

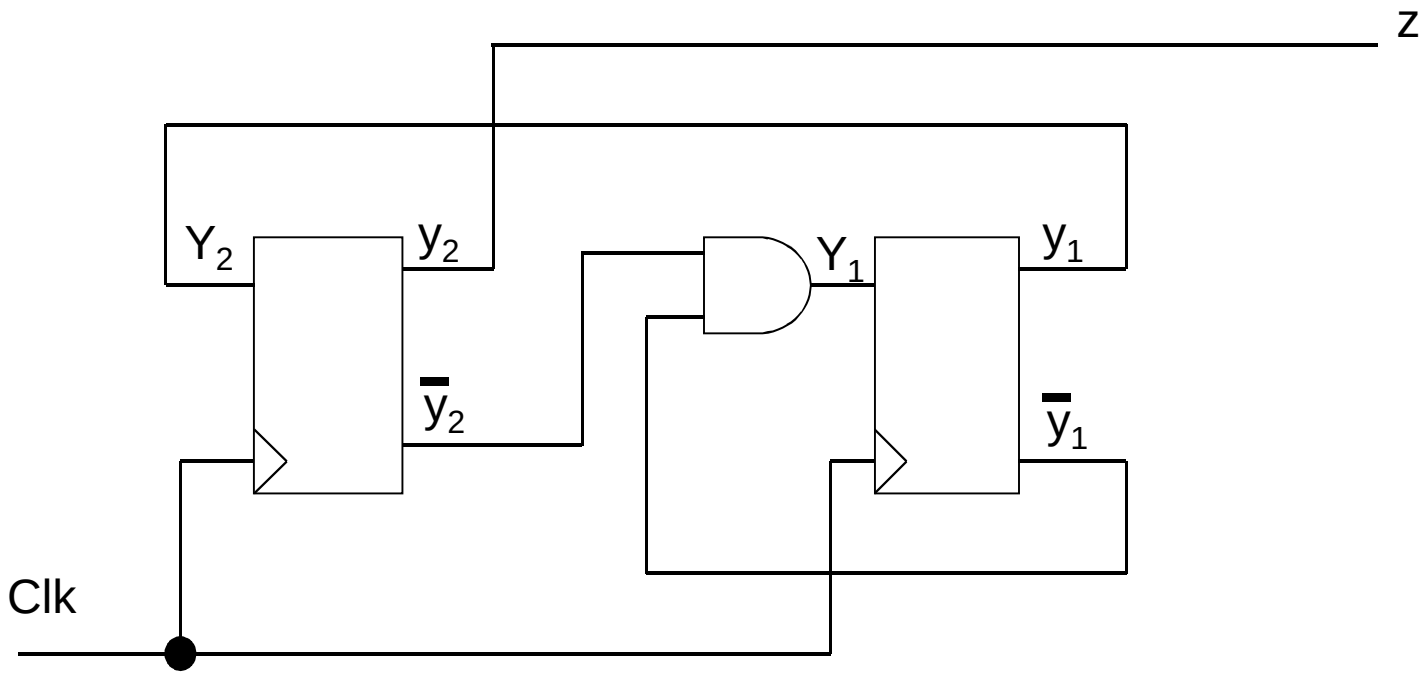
The unused state goes to S2



(0,0,1) sequence generator

Logic circuit for the sequence

The implementation uses D flip-flops



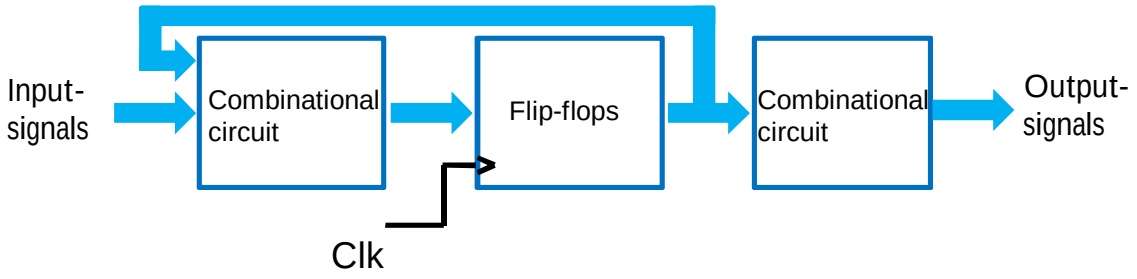
$Y_2 = y_1$ $\quad Y_1 = \overline{y}_2\overline{y}_1$ $\quad z = y_2$

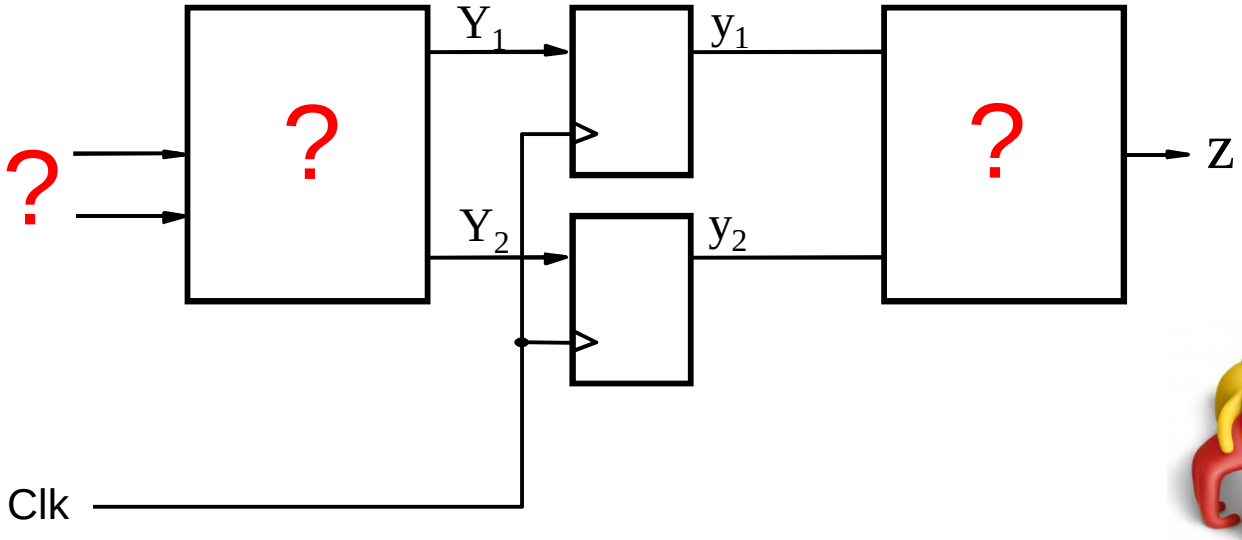
Logic circuit in the structure of Moore machine

$Y_1 = \bar{y}_2\bar{y}_1$

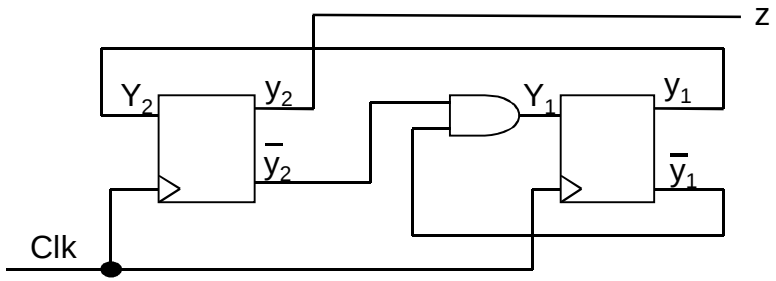
$Y_2 = y_1$

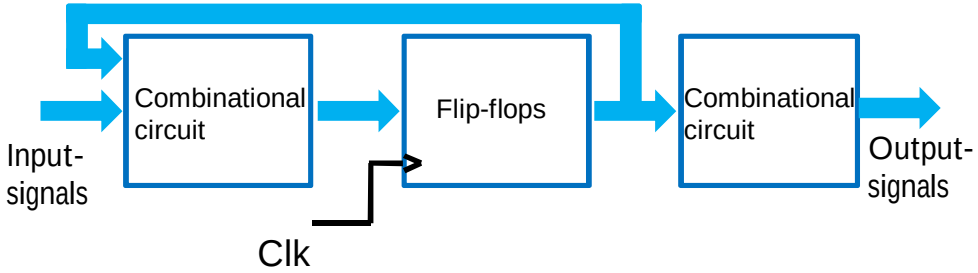
$z = y_2$





Logic circuit for the sequence

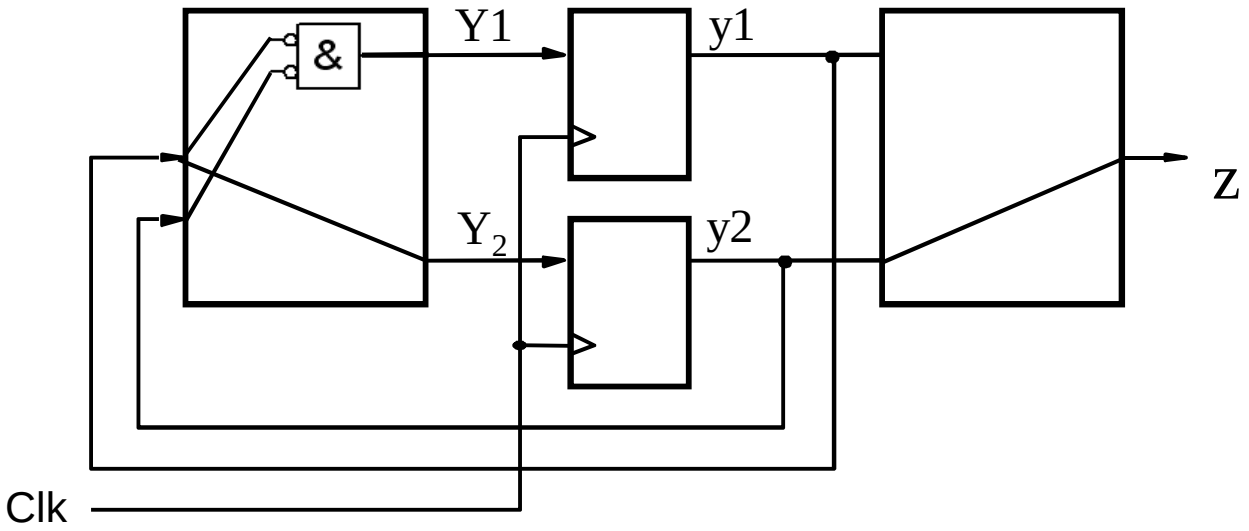




$Y_1 = \bar{y}_2\bar{y}_1$

$Y_2 = y_1$

$z = y_2$



State minimization

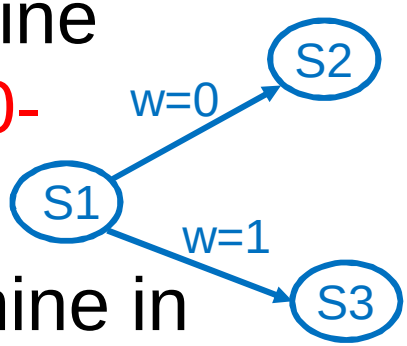
- When designing complex state machines, it often happens that there are equivalent states that can be grouped together to obtain a more efficient implementation
- Two states S_1 are S_2 are called equivalent if and only if, for every possible input sequence, the same output will be produced regardless of whether S_1 or S_2 is the initial state

State minimization

- The following example illustrates one minimization method which can be used for state minimization
- This method identifies states which are not equivalent (this is often easier)
- First, we introduce some terminology

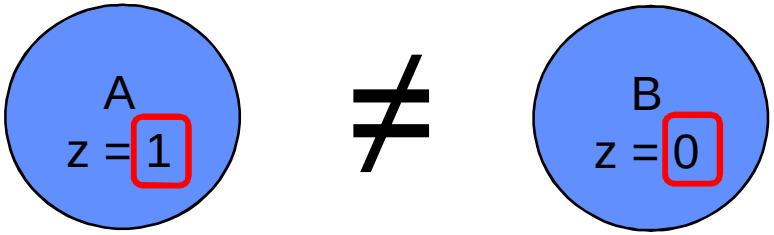
0- and 1-successors

- If input w = 0 is applied to a state machine in state S_1 and the result is that the machine moves to state S_2, we say that S_2 is a 0-successor of S_1
- If input w = 1 is applied to a state machine in state S_1 and the result is that the machine moves to state S_3, we say that S_3 is a 1-successor of S_1
- We will refer to successors as k-successors, where k can be 0 or 1



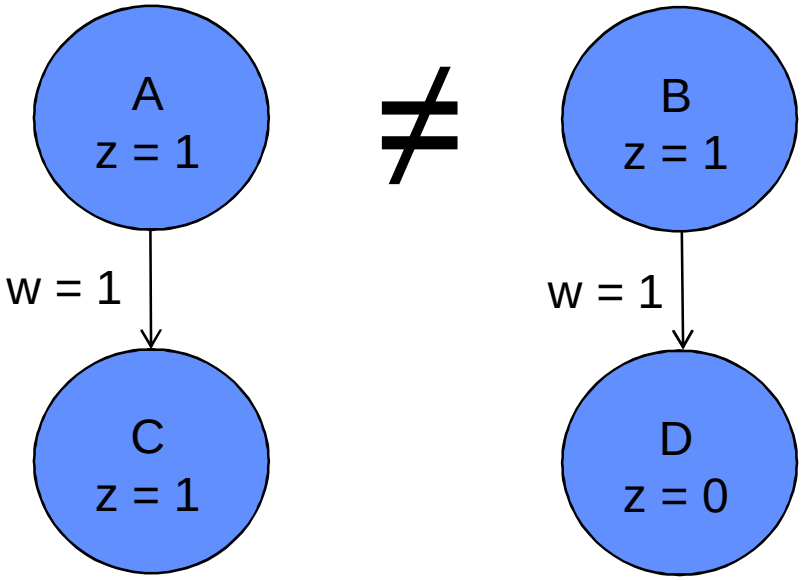
State minimization
Basic idea

- Two states are not equivalent if they have different output values

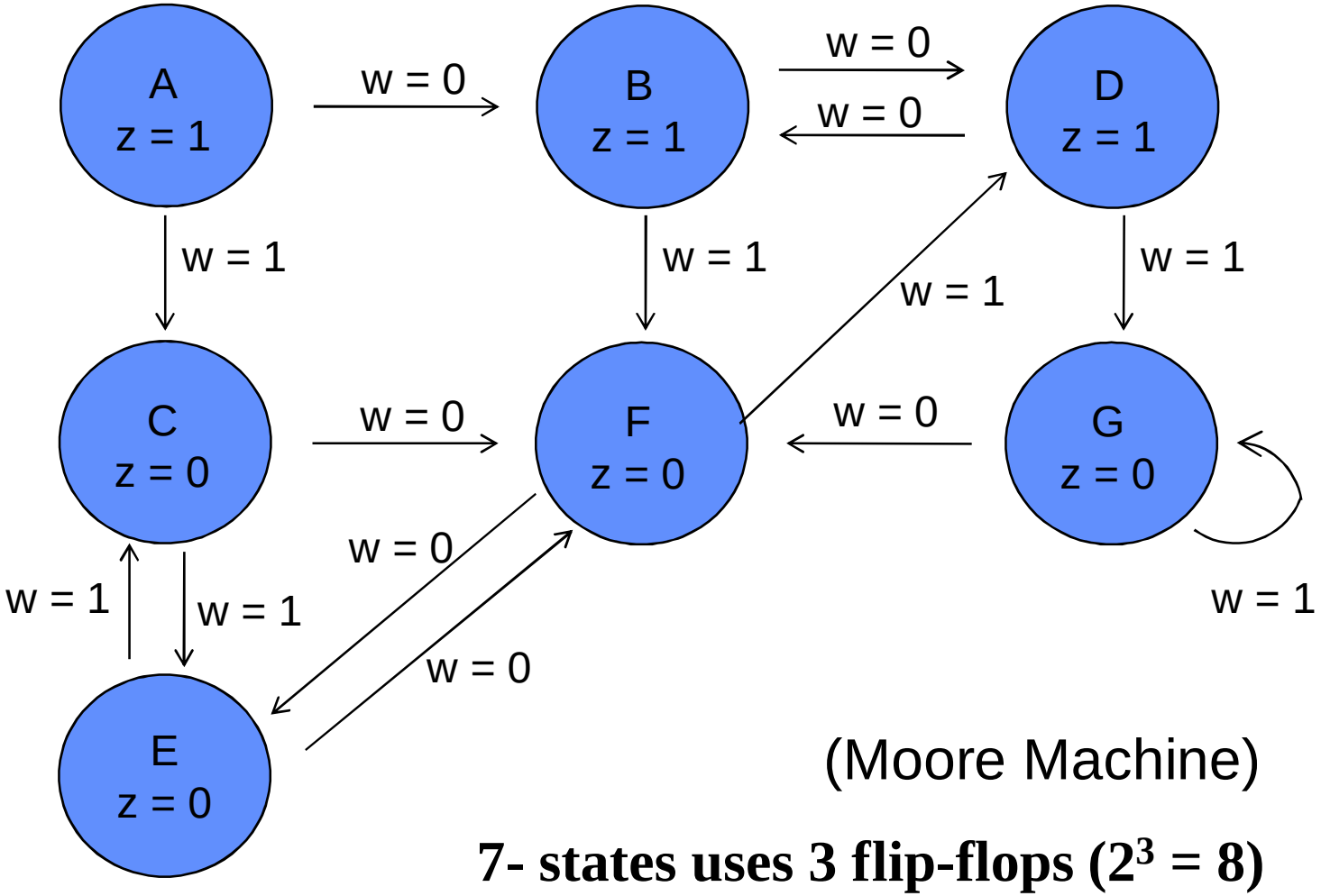


State minimization
Basic idea

- Two states are not equivalent if at least one of their k-successors are not equivalent



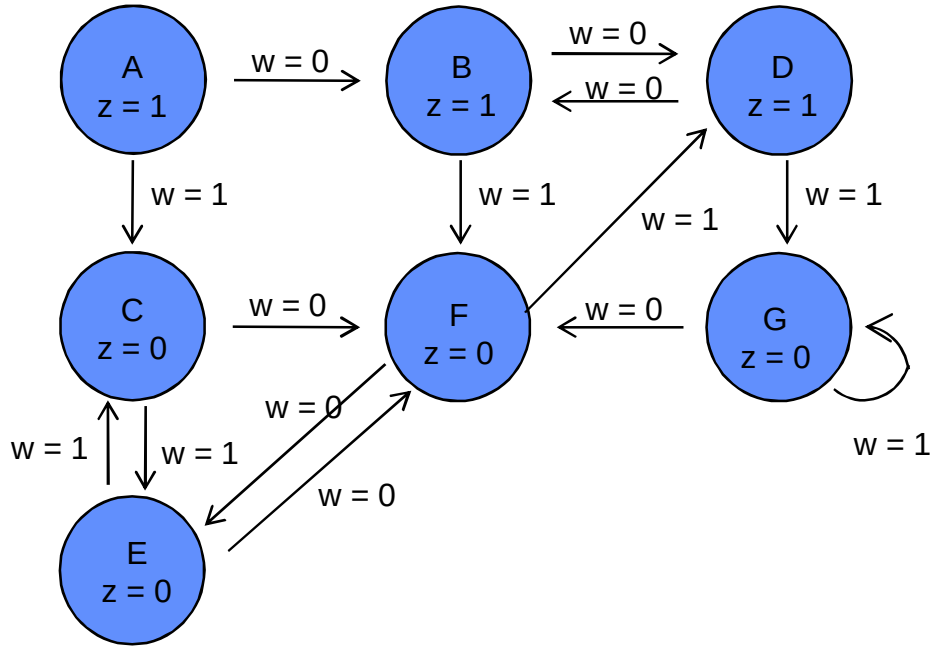
Example
State minimization



(Moore Machine)

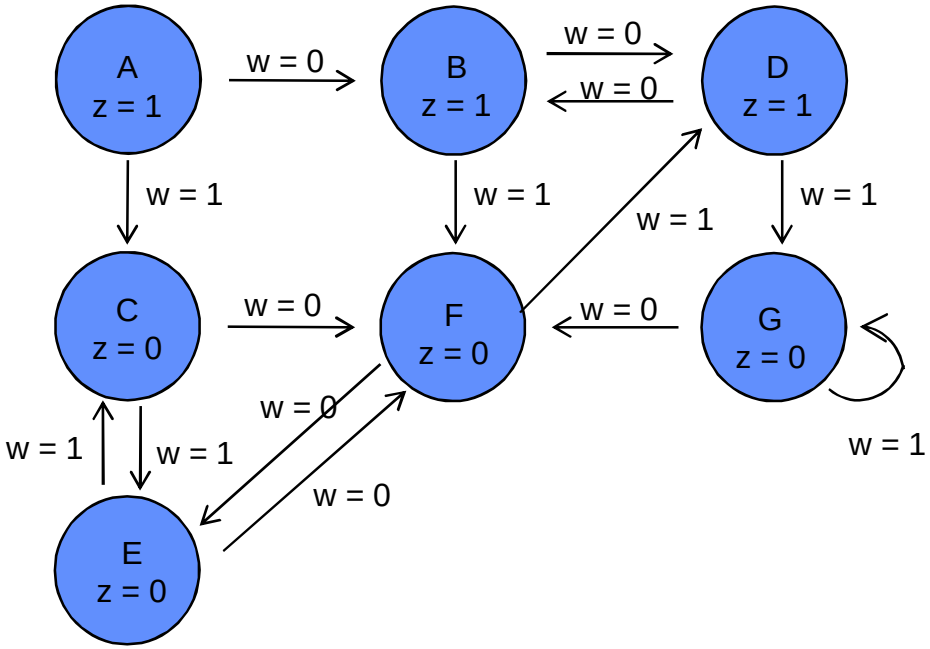
7- states uses 3 flip-flops ($2^3 = 8$)

State table



Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	B	C	1

State table



Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Partition

- The minimization procedure first considers the states of a machine as a set and then breaks this set into partitions that are not equivalent.
- A partition consists of one or more blocks
  - each block contains states that may be equivalent
  - different blocks contain states that are definitely not equivalent

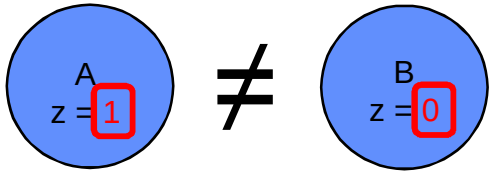
Example state minimization

- Start
  - Just one block containing all states
    - P_1= (ABCDEFG)

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Example state minimization P_1= (ABCDEFG)

- Stage 1:
  - Which states have different outputs?
    - ABD has output z = 1
    - CEFG have output z = 0
    - => P_2= (ABD) (CEFG)



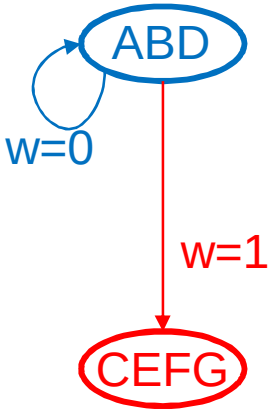
Present state	Next state $w = 0$	Next state $w = 1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

States A, B, D can therefore never be equivalent to any of the conditions C, E, F, G so they form different groups

Example state minimization
P_2= (ABD) (CEFG)

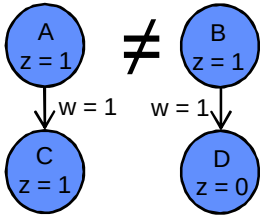
Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

- Stage 2
  - Which states have different k-successors?



- Block ABD
  - 0-successor: A → B, B → D, D → B (all transitions go to the same block)
  - 1-successor: A → C, B → F, D → G (all transitions go to the same block)
- Block CEFG
  - 0-successor: C → F , E → F , F → E , G → F (all transitions go to the same block)
  - 1-successor: C → E , E → C , F → D , G → G (F → D goes to another block)

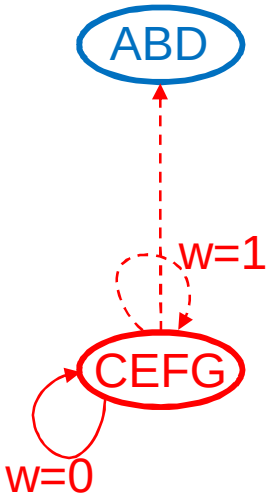
=> **P_3= (ABD) (CEG) (F)**



Example state minimization
P_2= (ABD) (CEFG)

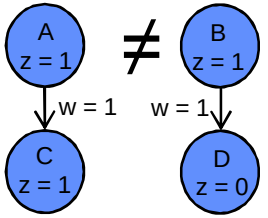
Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

- Stage 2
  - Which states have different k-successors?



- Block ABD
  - 0-successor: A → B , B → D , D → B (all transitions go to the same block)
  - 1-successor: A → C , B → F , D → G (all transitions go to the same block)
- Block CEFG
  - 0-successor: C → F, E → F, F → E, G → F (all transitions go to the same block)
  - 1-successor: C → E , E → C, F → D, G → G (F → D goes to another block)

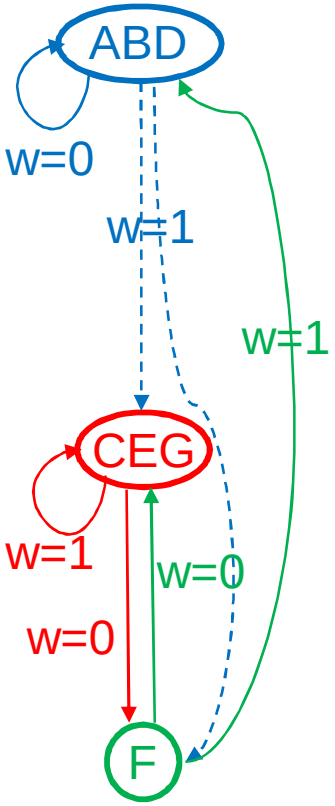
=> P_3= (ABD) (CEG) (F)



Example state minimization
P3= (ABD) (CEG) (F)

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

- Step 3
  - What states have different k-successors?



- Block ABD
  - 0-successor: A → B, B → D, D → B (all transitions go to the same block)
  - 1-successor: A → C, B → F, D → G (B → F goes to another block)

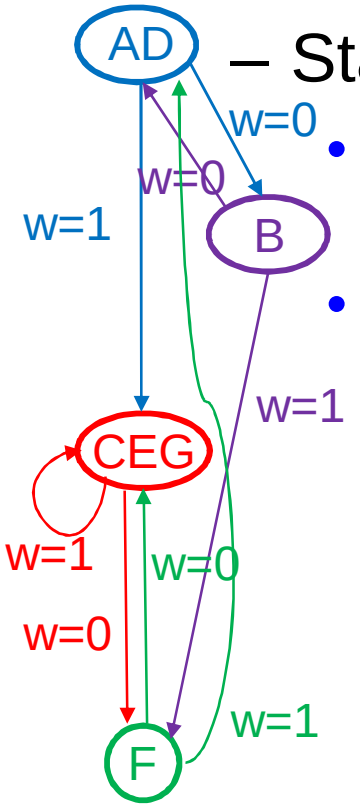
  => **P4= (AD) (B) (CEG) (F)**
- Block (CEG)
  - 0-successor: C → F, E → F, G → F (all transitions go to the same block)
  - 1-successor: C → E, E → C, G → G (all transitions go to the same block)

  => **P4= (AD) (B) (CEG) (F)**

Example state minimization
P4= (AD) (B) (CEG) (F)

Present state	Next state $w=0$	Next state $w=1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

- Next partition $\mathbf{P_5}$ becomes the same as $\mathbf{P_4}$. Thus the procedure is finished.
  - – States in each block are equivalent
    - if they were not, their k-successors would have to be in different blocks
    - A becomes the representive of AD and C represents CEG.



$$\mathbf{P_4} = (\mathrm{AD})(\mathrm{B})(\mathrm{CEG})(\mathrm{F}) = (\mathrm{A})(\mathrm{B})(\mathrm{C})(\mathrm{F})$$

Final state table

Present state	Next state $w = 0$	Next state $w = 1$	Output z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

Present state	Next State w = 0	Next State w = 1	Output z
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Final State Table

$$\mathbf{P_4} = (AD)(B)(CEG)(F) = (A)(B)(C)(F)$$

Final state table

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

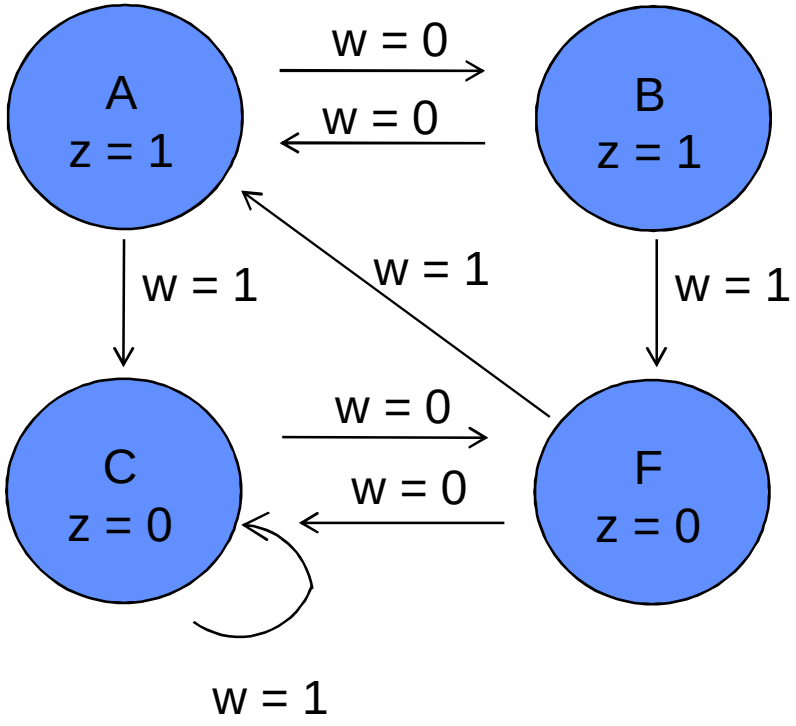
Present state	Next State		Output
	w = 0	w = 1	z
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Final State Table

$$\mathbf{P_4} = (AD)(B)(CEG)(F) = (A)(B)(C)(F)$$

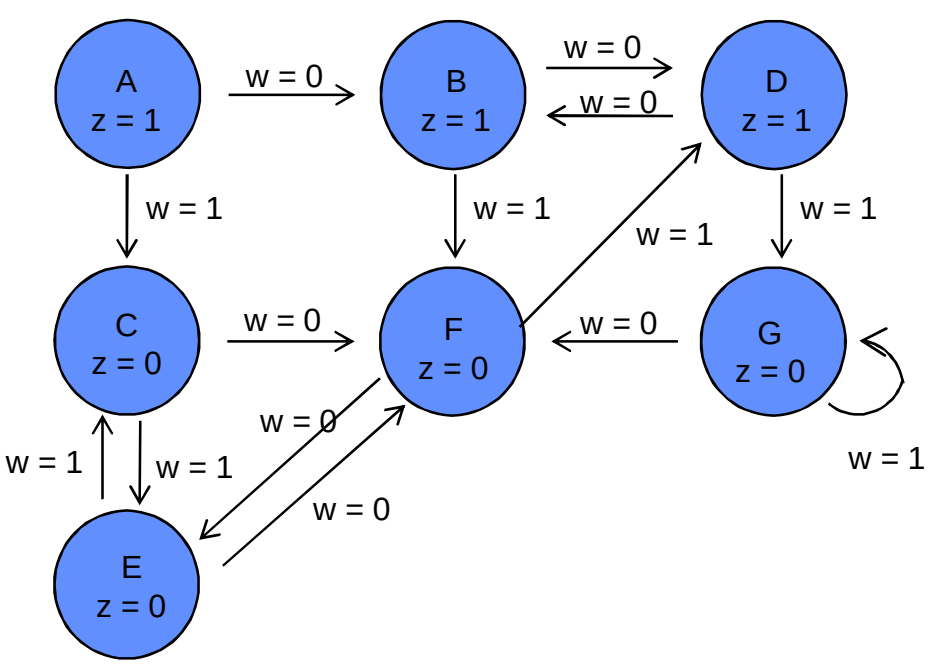
Final state diagram

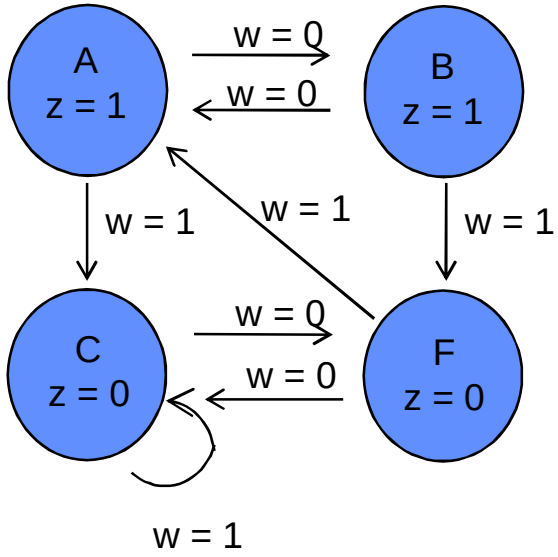
Present state	Next State		Output z
	w = 0	w = 1	
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0



4 states needs 2 flip-flops ($2^2 = 4$).

Comparison





- Only 2 flip-flops are needed to implement 4 states in the minimized state table
- 3 flip-flops are needed to implement 7 states in the original state table

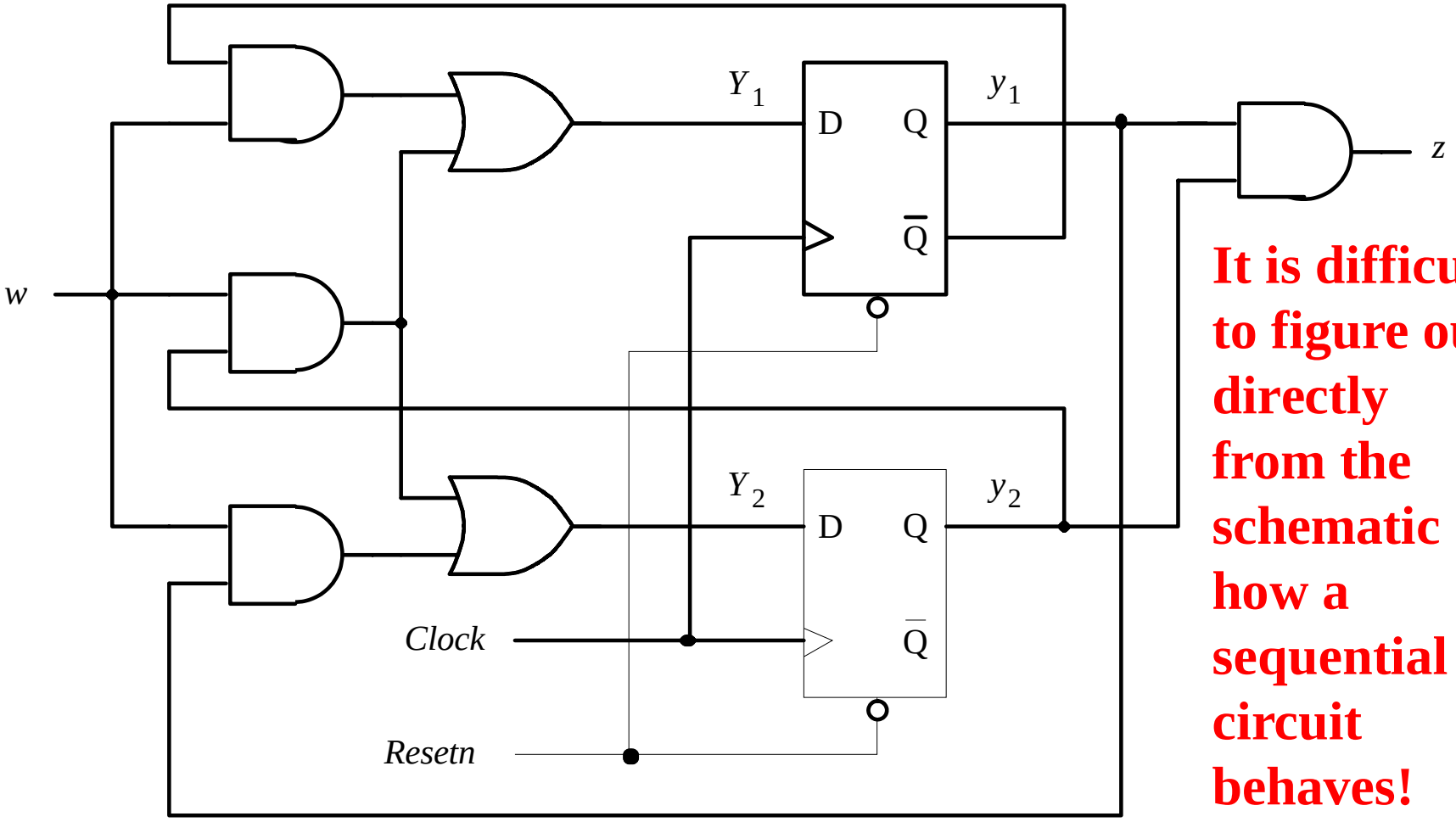
Some thought!

- Fewer state does not necessarily lead to a simpler design!
  - The advantage of state minimization is instead that it makes it easier to create the initial state diagram, when you do not have to get it to be minimal from the beginning!

Analysis of synchronous sequential circuits

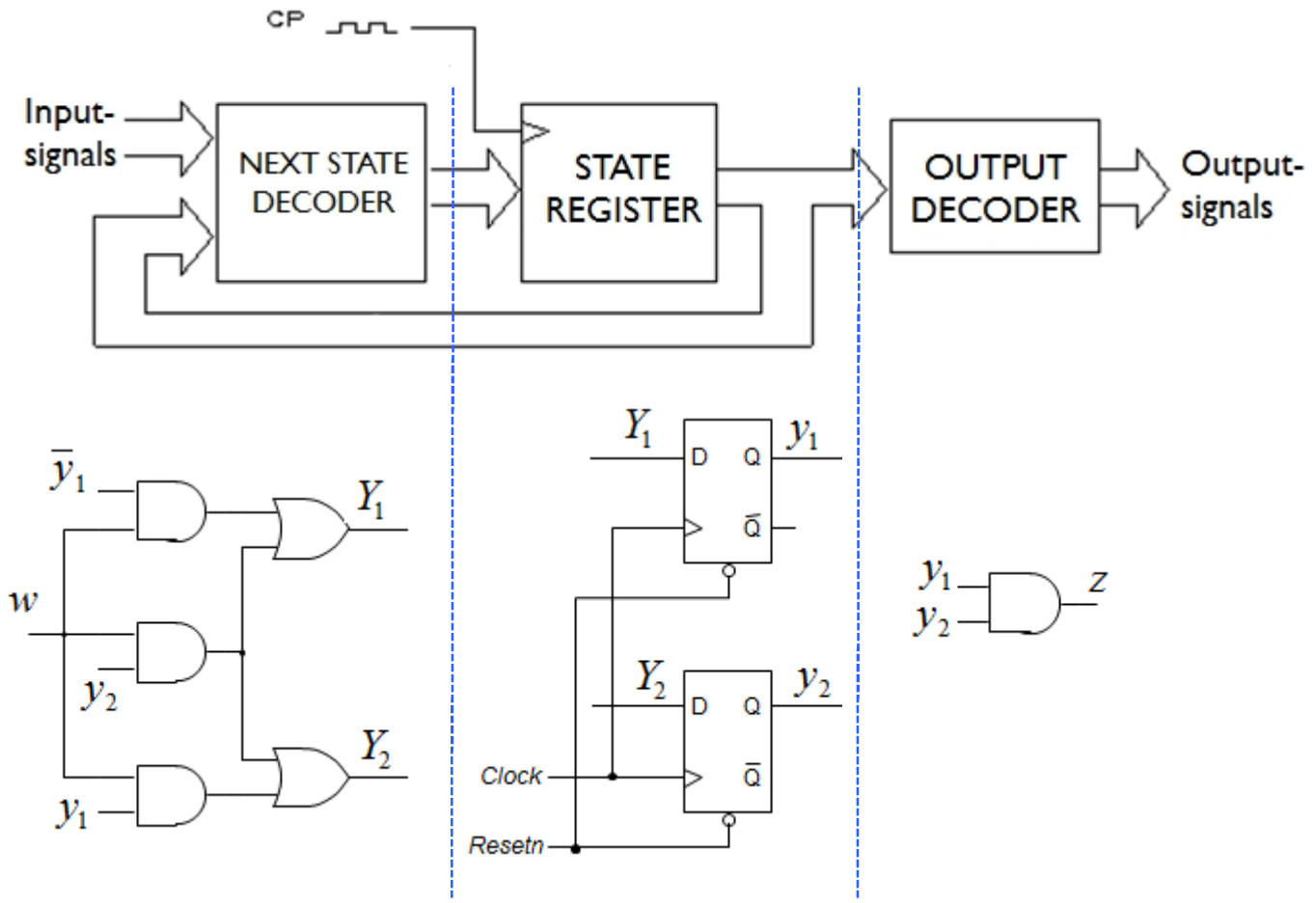
- Given an implementation of a synchronous circuit, we can produce its function by making the synthesis steps in a reverse order!
  1. Get expressions for
     - next state decoder
     - output decoder
  2. Get the state table
  3. Draw the state diagram

Example: Analysis of a synchronous sequential circuit



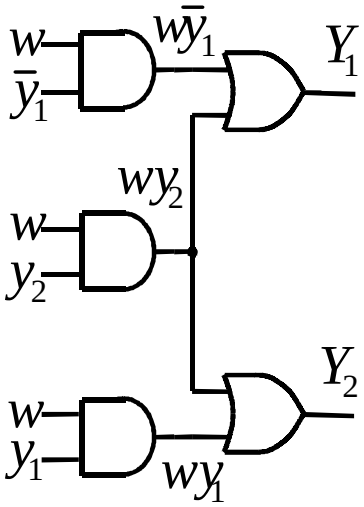
It is difficult to figure out directly from the schematic how a sequential circuit behaves!

Example: Moore-machine!



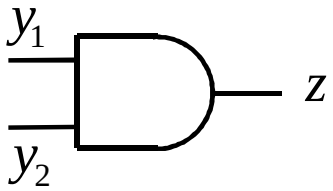
Example: Analysis of a synchronous sequential circuit

1. Get expressions for
   - next state decoder
   - output decoder



$$Y_1 = w\bar{y}_1 + wy_2$$

$$Y_2 = wy_1 + wy_2$$



$$z = y_1 y_2$$

Example: Analysis of a synchronous sequential circuit

2. Get the state table

Present state y_2y_1	Next State $w=0$ Y_2Y_1	Next State $w=1$ Y_2Y_1	Output z
0 0	0 0	01	0
0 1	0 0	10	0
1 0	0 0	11	0
1 1	0 0	11	1

A:"00" B:"01" C:"10" D:"11"

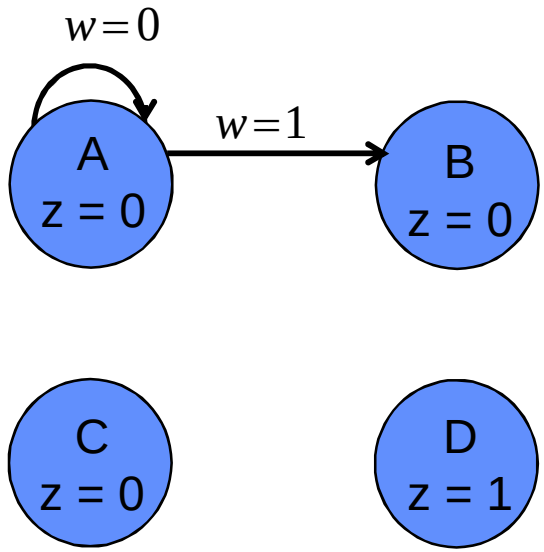
Present state	Next state $w=0$	Next state $w=1$	Output z
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

$$Y_2 = wy_1 + wy_2 \qquad Y_1 = w\bar{y}_1 + wy_2 \qquad z = y_1 \cdot y_2$$

Example: Analysis of a synchronous sequential circuit

3. Draw state diagram
   – Left as exercise for students... (but check the state table to make sure it is a bit sequence detector for three subsequent 1s)

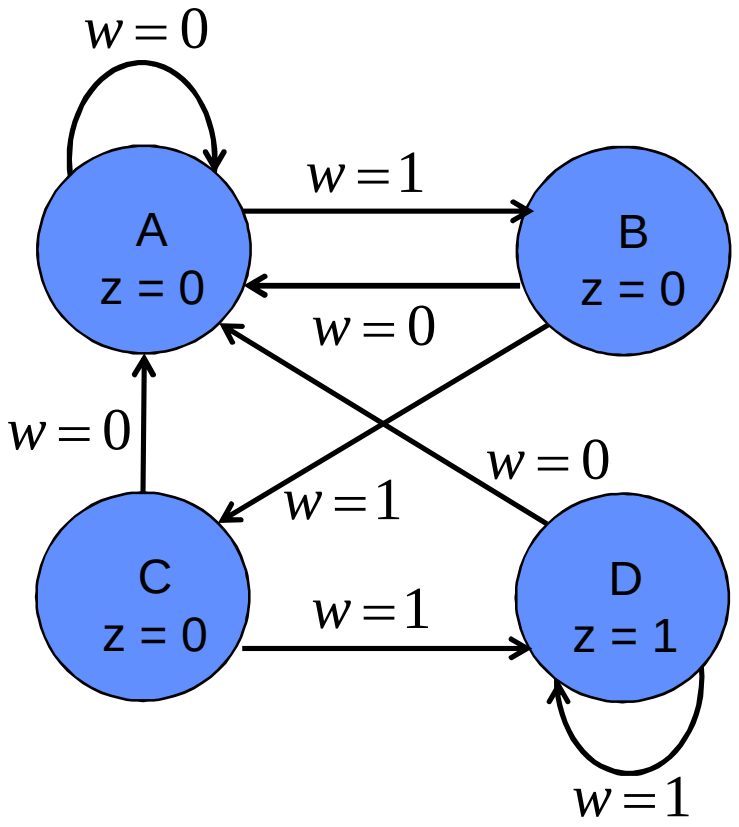
Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1



State diagram

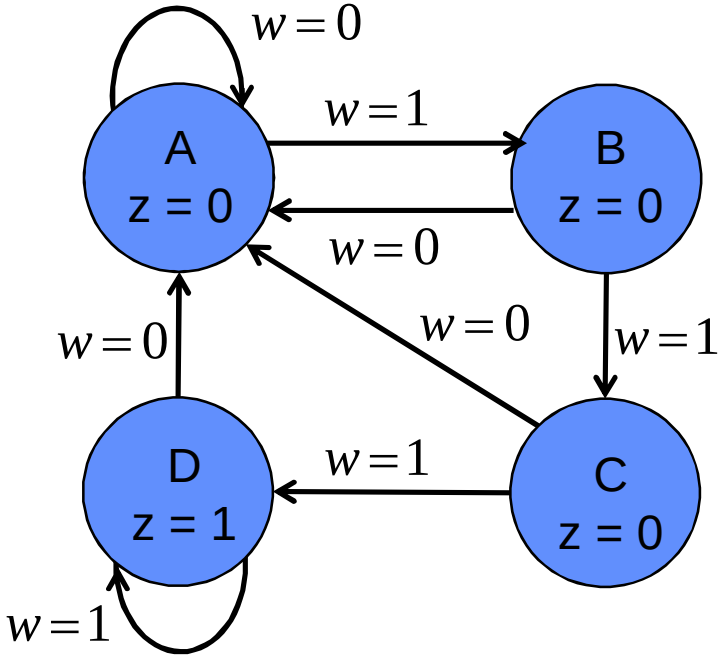
Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

Sometimes you may need to change the order of the states to get a clearer chart.



State diagram

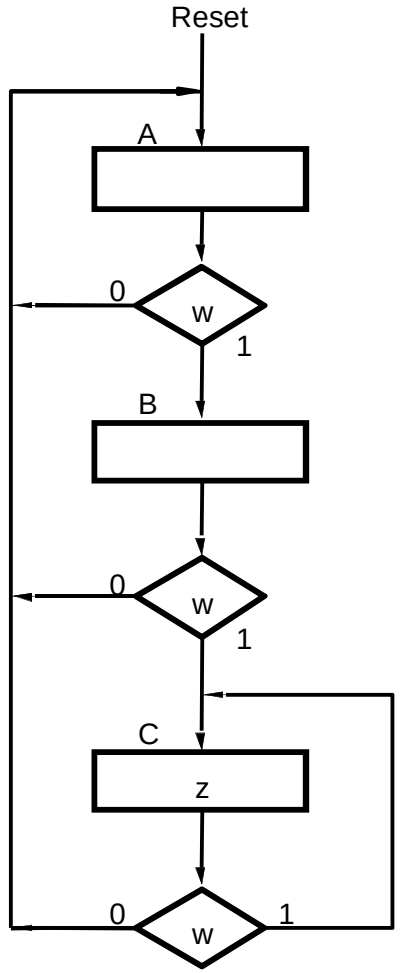
Present state	Next state		Output z
	w = 0	w = 1	
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1



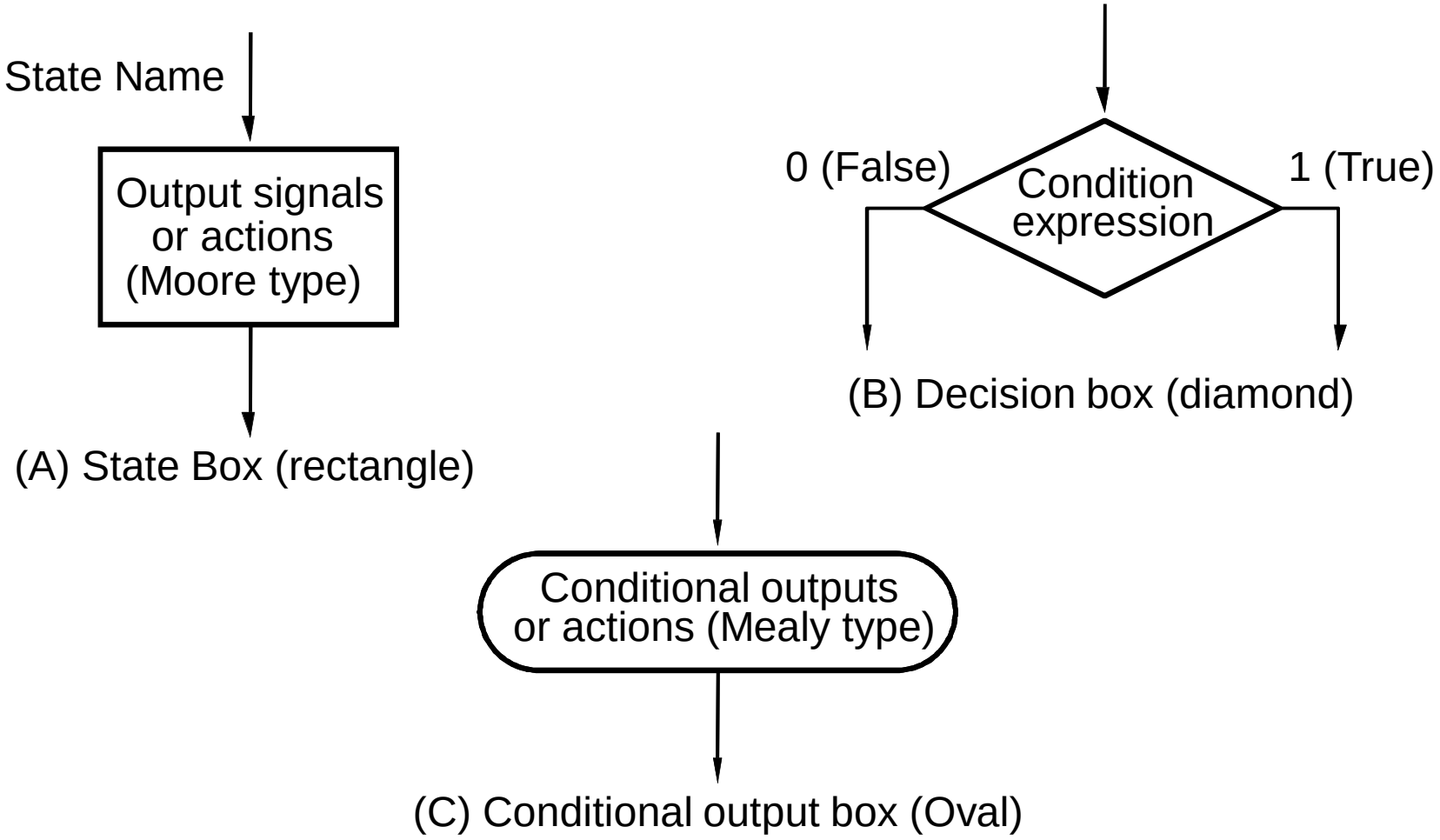
C and D have changed places resulting in no intersecting state arrows.

ASM Charts

- State transition diagrams are convenient for describing the behavior of small state machines only
- To describe larger state machines, another type of diagrams, called *Algorithmic State Machine (ASM)* charts are often used
- An ASM is a flow diagram consisting of three types of elements: state box, decision box and conditional output box



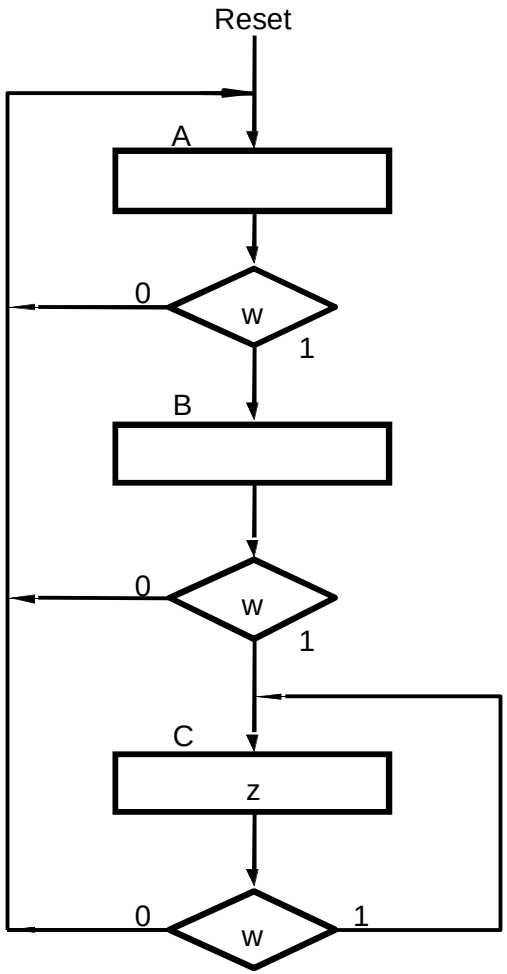
ASM Charts

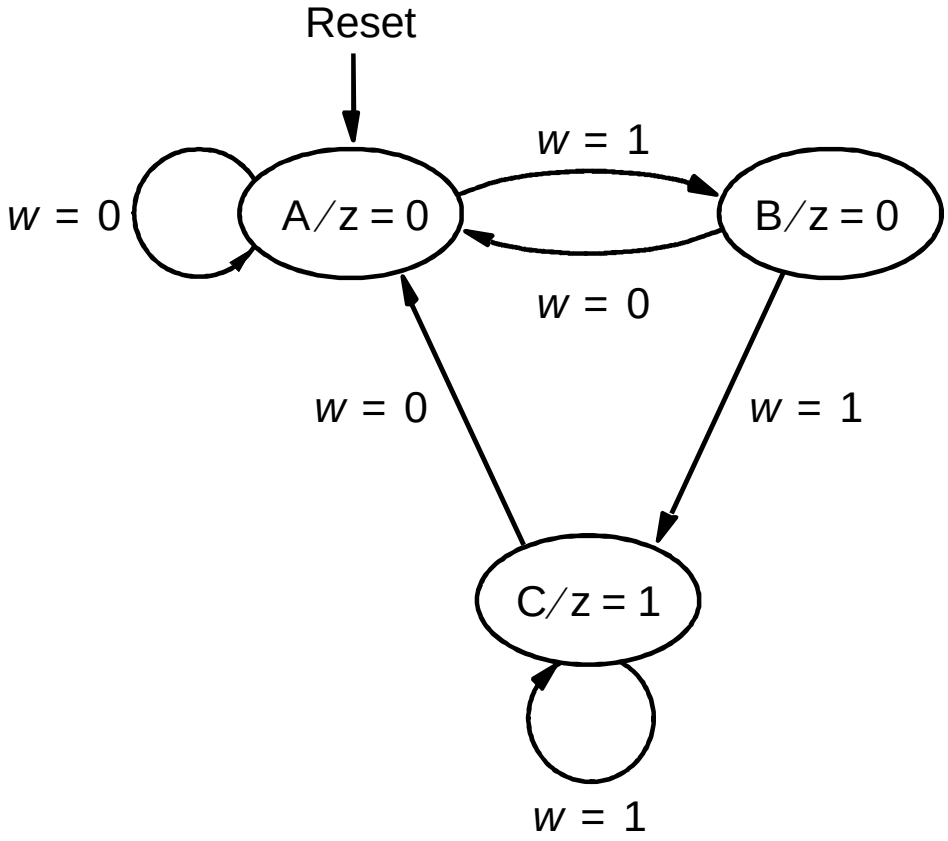

State Name
Output signals or actions (Moore type)
(A) State Box (rectangle)
0 (False)
Condition expression
1 (True)
(B) Decision box (diamond)
Conditional outputs or actions (Mealy type)
(C) Conditional output box (Oval)

ASM Charts

- State Box
  - Represents a state in a FSM
    - Output values for state are given here (*Moore outputs*)
- Decision Box
  - Depending on the values of the input signals, it determines a transition to the next state
- Conditional outputs Box
  - Specifies the values of the outputs at a state transition (*Mealy outputs)*

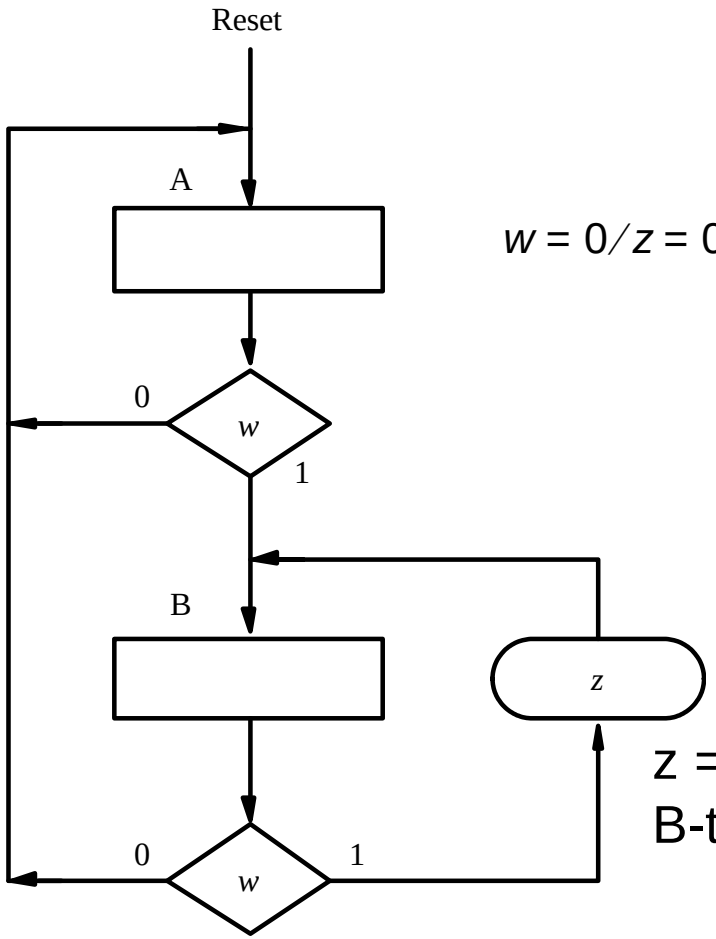
ASM chart for 11 sequence detector (Moore)

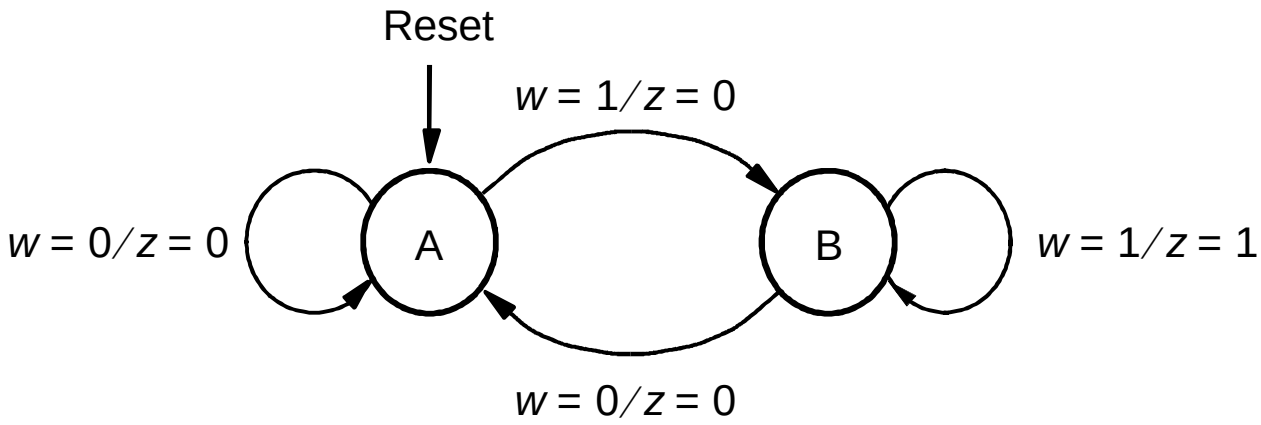




z = 1 only in the state C

ASM chart for 11 sequence detector (Mealy)



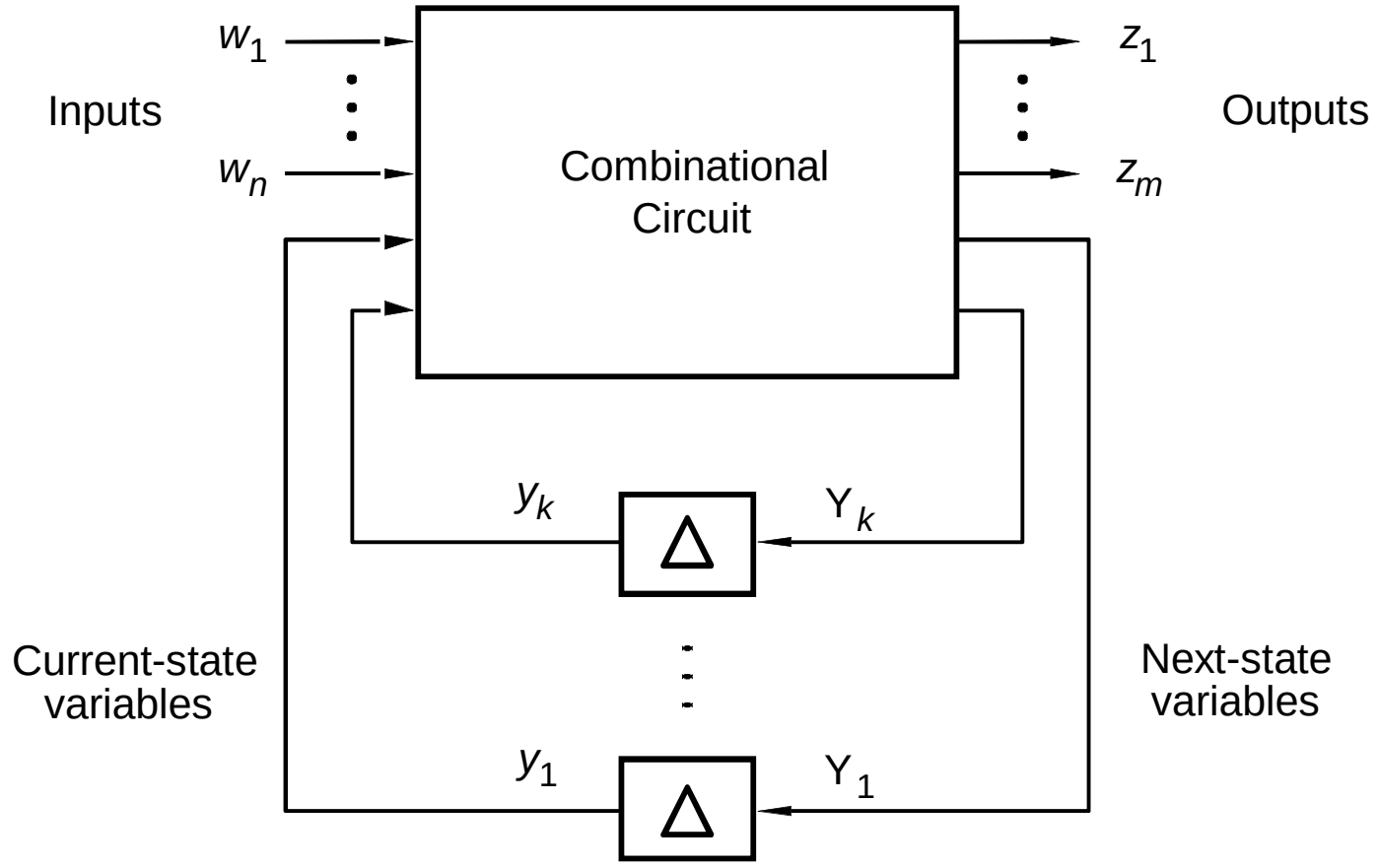


z = 1 only when the state transition B-to-B with w = 1 takes place

Formal model for sequential circuits

- To treat state machines in a formal way, we need a formal model
- The following model can describe both Moore and Mealy machines

Formal model for sequential circuits



Formal model for sequential circuits

- A synchronous sequential circuit can be formally defined as

$$M = (W, Z, S, \varphi, \lambda)$$

- W, Z, and S are finite, nonempty sets of inputs, outputs and states, respectively
- φ is the state transition function, such as S(t+1) = φ[W(t), S(t)]
- λ is the output function, such as λ(t) = λ(S(t)) for the Moore model and λ(t) = λ(W(t), S(t)) for the Mealy model

Formal model for sequential circuits

$$M = (W, Z, S, \varphi, \lambda)$$

$$W_{inputs} = \{w_1, w_2, ..., w_m\}$$

$$Z_{outputs} = \{z_1, z_2, ..., z_m\}$$

$$S_{states} = \{S_1, S_2, ..., S_m\}$$

$$y_{present-state-variables} = \{y_1, y_2, ..., y_m\}$$

$$Y_{next-state-variables} = \{Y_1, Y_2, ..., Y_m\}$$

$$S(t + \Delta t) = Y_1 ... Y_m = \varphi(W(t), S(t))$$

$$\lambda_{Moore}(t) = \lambda(S(t))$$

$$\lambda_{Mealy}(t) = \lambda(W(t), S(t))$$

Summary

- State minimization
- Analysis of a synchronous sequential circuit
- ASM charts
- Formal model for sequential circuits
- Next lecture: BV pp. 98-118, 418-426, 508-519